CPSC 440: Advanced Machine Learning

Restricted Boltzmann Machines

Mark Schmidt

University of British Columbia

Winter 2021

Last Time: Learning in UGMs

- We discussed log-linear parameterization of UGMs,

$$\phi_j(s) = \exp(w_{j,s}), \quad \phi_{jk}(s,s') = \exp(w_{j,k,s,s'}), \quad \phi_{jkl}(s,s',s'') = \exp(w_{j,k,l,s,s',s''}).$$

  the likelihood of an example x given parameter w is given by

$$p(x \mid w) = \frac{\exp\left(w^T F(x)\right)}{Z},$$

  and the feature functions $F(x)$ count the number of times we use each w_j.

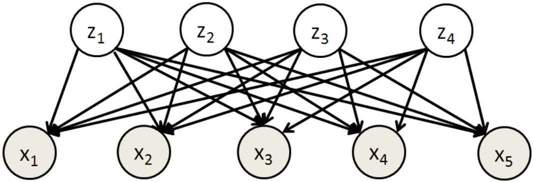
- Gradient of the NLL with respect to a particular w_j has the form

$$\nabla_{w_{10,3}} f(w) = \underbrace{-\frac{1}{n}\left[\sum_{i=1}^{n} I[x_{10}^i = 3]\right]}_{\text{frequency in data}} + \underbrace{p(x_{10} = 3 \mid w)}_{\text{model ``frequency''}}.$$

- There are different ways to address the annoying term:
  - For example, run Gibbs sampling to approximate it with Monte Carlo.

Last Time: Latent DAG Model

- Last time we discussed the following model:

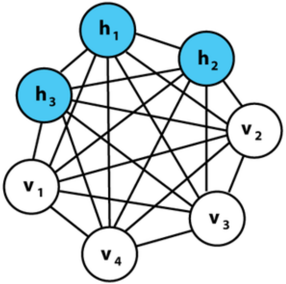


- With k hidden binary nodes, a mixture model with 2^k clusters.
  - You can think of each z_c as a "part" that can be included or not ("binary PCA").

- Usually assume $p(x_j \mid z_1, z_2, z_3, z_4)$ is a linear model (Gaussian, logistic, etc.).
  - With d visible x_j and k hidden z_j, we only have dk parameters.
- Unfortunately, somewhat hard to use:
  - Combinatorial "explaining away" between z_c value when conditioning on x.
  - Restricted Boltzmann Machines (RBMs) are a similar undirected model...

Boltzmann Machines

- Boltzmann machines are UGMs with binary latent variables:

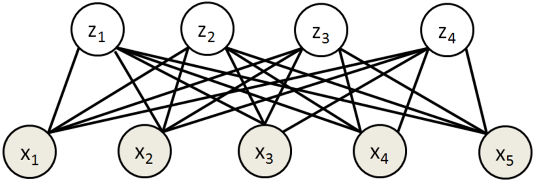


https://en.wikipedia.org/wiki/Boltzmann_machine

- Yet another latent-variable model for density estimation.
  - Hidden variables again give a combinatorial latent representation.
- Hard to do anything in this model, even if you know all the z (or x).

Restricted Boltzmann Machine

- By restricting graph structure, some things get easier:
  - Restricted Boltzmann machines (RBMs): edges only between the x_j and z_c.



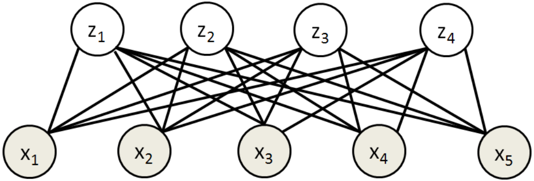
- Bipartite structure allows block Gibbs sampling given one type of variable:
  - Conditional UGM is disconnected.

- Given visible x, we can sample each z_c independently.
- Given hidden z, we can sample each x_j independently.

Restricted Boltzmann Machines

- The RBM graph structure leads to a joint distribution of the form

$$p(x,z) = \frac{1}{Z}\left(\prod_{j=1}^{d}\phi_j(x_j)\right)\left(\prod_{c=1}^{k}\phi_c(z_c)\right)\left(\prod_{j=1}^{d}\prod_{c=1}^{k}\phi_{jc}(x_j,z_c)\right).$$



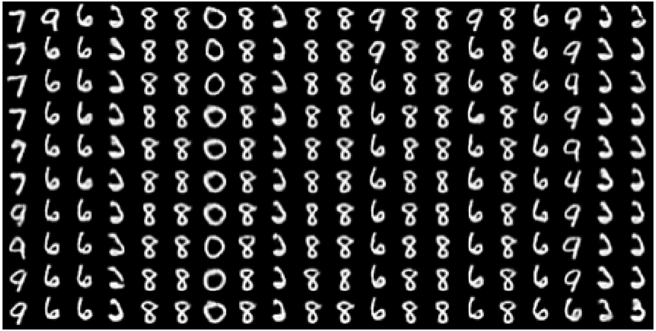
- RBMs usually use a log-linear parameterization like

$$p(x,z) \propto \exp\left(\sum_{j=1}^{d}x_jw_j + \sum_{c=1}^{k}z_cv_c + \sum_{j=1}^{d}\sum_{c=1}^{k}x_jw_{jc}z_c\right),$$

for parameters w_j, v_c, and w_{jc} (first term would be different for continuous x_j).

Generating Digits with RBMs

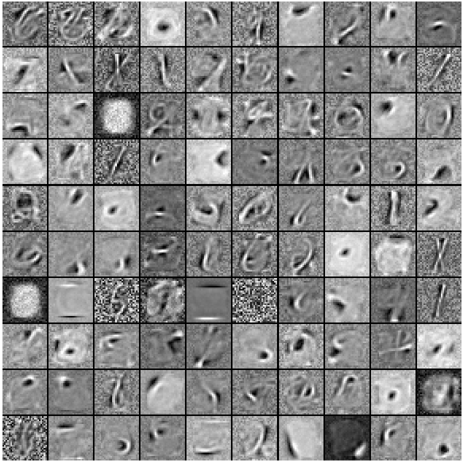
Here are the samples generated by the RBM after training. Each row represents a mini-batch of negative particles (samples from independent Gibbs chains). 1000 steps of Gibbs sampling were taken between each of those rows.



http://deeplearning.net/tutorial/rbm.html

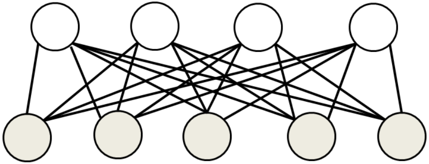
Generating Digits with RBMs

Visualizing each z_c's interaction parameters (w_{jc} for all j) as images:



Learning UGMs with Hidden Variables

- For RBMs we have hidden variables:



- With hidden ("nuissance") variables z the observed likelihood has the form

$$p(x) = \sum_z p(x,z) = \sum_z \frac{\tilde{p}(x,z)}{Z}$$
$$= \frac{1}{Z} \underbrace{\sum_z \tilde{p}(x,z)}_{Z(x)} = \frac{Z(x)}{Z},$$

where $Z(x)$ is the partition function of the conditional UGM given x.
  - $Z(x)$ is cheap in RBMs because the z are independent given x.

Learning UGMs with Hidden Variables

- This gives an observed NLL of the form
$$-\log p(x) = -\log(Z(x)) + \log Z,$$
where $Z(x)$ sums over hidden z values, and Z sums over z and x.

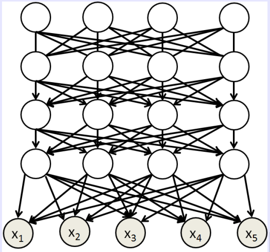
- The second term is convex but the first term is non-convex.
  - This is expected when we have hidden variables.

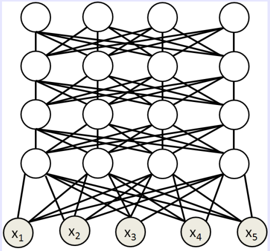
- With a log-linear parameterization, the gradient has the form
$$-\nabla \log p(x) = -\mathbb{E}_{z \mid x}[F(X,Z)] + \mathbb{E}_{z,x}[F(X,Z)].$$

- For RBMs, first term is cheap due to independence of z given x.
- We can approximate second term using block Gibbs sampling.
  - For other problems, you would also need to approximate first term.

Deep Belief Networks and Deep Boltzmann Machines

- Around 15 years ago, a hot topic was "stacking" latent DAGs and/or RBMs:
  - Part of the motivation for peope to re-consider "deep" models.
  - These architectures were popular because they were deep but nice for sampling.
    - And it was common to use "train on RBM" as an ingredient for learning.

Deep belief net: 

Deep Boltzmann: 

- Post-lecture bonus slides go through some of the details if you are interested.
  - `https://www.youtube.com/watch?v=KuPai0ogiHk`

Outline

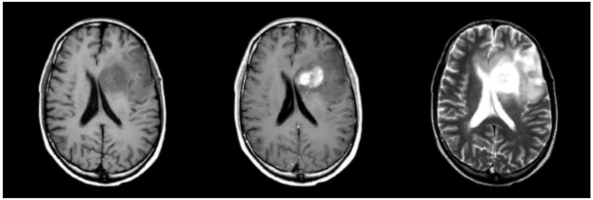
3 Classes of Structured Prediction Methods

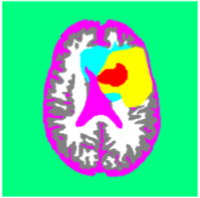
3 main approaches to structured prediction (predicting object y given features x):

1. Generative models use $p(y \mid x) \propto p(y, x)$ as in naive Bayes.
   - Turns structured prediction into density estimation.
     - But remember how hard it was just to model images of digits?
     - We have to model features and solve supervised learning problem.
2. Discriminative models directly fit $p(y \mid x)$ as in logistic regression (next topic).
   - View structured prediction as conditional density estimation.
     - Just focuses on modeling y given x, not trying to model features x.
     - Lets you use complicated features x that make the task easier.
3. Discriminant functions just try to map from x to y as in SVMs.
   - Now you don't even need to worry about calibrated probabilities.

Motivation: Automatic Brain Tumor Segmentation

- Task: identification of tumours in multi-modal MRI.





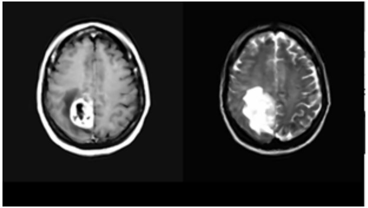
- Applications:
  - Radiation therapy target planning, quantifying treatment response.
  - Mining growth patterns, image-guided surgery.
- Challenges:
  - Variety of tumor appearances, similarity to normal tissue.
  - "You are never going to solve this problem".

Brain Tumour Segmentation with Label Dependencies

- After a lot pre-processing and feature engineering (convolutions, priors, etc.), final system used logistic regression to label each pixel as "tumour" or not.

$$p(y_c \mid x_c) = \frac{1}{1 + \exp(-2y_c w^T x_c)} = \frac{\exp(y_c w^T x_c)}{\exp(w^T x_c) + \exp(-w^T x_c)}$$

- Gives a high "pixel-level" accuracy, but sometimes gives silly results:



- Classifying each pixel independently misses dependence in labels y^i:
  - We prefer neighbouring voxels to have the same value.

Brain Tumour Segmentation with Label Dependencies

- With independent logistic, conditional distribution over all labels in one image is

$$p(y_1, y_2, \dots, y_k \mid x_1, x_2, \dots, x_k) = \prod_{c=1}^{k} \frac{\exp(y_c w^T x_c)}{\exp(w^T x_c) + \exp(-w^T x_c)}$$
$$\propto \exp\left(\sum_{c=1}^{d} y_c w^T x_c\right),$$

where here x_c is the feature vector for position c in the image.

- We can view this as a log-linear UGM with no edges,

$$\phi_c(y_c) = \exp(y_c w^T x_c),$$

so given the x_c there is no dependence between the y_c.

Brain Tumour Segmentation with Label Dependencies

- Adding an Ising-like term to model dependencies between y_i gives

$$p(y_1, y_2, \dots, y_k \mid x_1, x_2, \dots, x_k) \propto \exp\left(\sum_{c=1}^{k} y_c w^T x_c + \sum_{(c,c') \in E} y_c y_{c'} v\right),$$

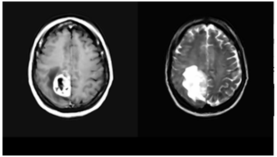
- Now we have the same "good" logistic regression model, but v controls how strongly we want neighbours to be the same.

- Note that we're going to jointly learn w and v.
  - We'll find the optimal joint logistic regression and Ising model.

- When we model conditional of y given x as a UGM, we call it a conditional random field (CRF).
  - Key advantadge of this (discriminative) approach:
    - Don't need to model features x as in "generative" models.
    - We saw with MNIST digits that modeling images is hard.

Conditional Random Fields for Segmentation

- Recall the performance with the independent classifier:



- The pairwise CRF better modelled the "guilt by association":
  - Trained with pseudo-likelihood. Added constraint $v \geq 0$ to use graph cut decoding.

(We were using edge features $x_{cc'}$ too, see bonus (and different λ on edges).)

- CRFs are like logistic regression (no modeling x) vs naive Bayes (modeling x).
  - $p(y \mid x)$ (discriminative) vs. $p(y, x)$ (generative).

Conditional Random Fields

- The brain CRF can be written as a conditional log-linear models,

$$p(y \mid x, w) = \frac{1}{Z(x)} \exp(w^T F(x, y)),$$

for some parameters w and features $F(x, y)$.

- The NLL is convex and has the form

$$-\log p(y \mid x, w) = -w^T F(x, y) + \log Z(x),$$

and the gradient can be written as

$$-\nabla \log p(y \mid x, w) = -F(x, y) + \mathbb{E}_{y \mid x}[F(x, y)].$$

- Unlike before, we now have a $Z(x)$ and set of expectations for each x.
  - Train using gradient methods like quasi-Newton or stochastic gradient.

Rain Data without Month Information

- Consider an Ising UGM model for the rain data with tied parameters,

$$p(y_1, y_2, \dots, y_k) \propto \exp\left(\sum_{c=1}^{k} y_c\omega + \sum_{c=2}^{k} y_c y_{c-1}\nu\right).$$

- First term reflects that "not rain" is more likely.
- Second term reflects that consecutive days are more likely to be the same.
  - This model is equivalent to a Markov chain model.

- We could condition on month to model "some months are less rainy".

Rain Data with Month Information using CRFs

- Discriminative appraoch: fit a CRF model conditioned on month x,

$$p(y_1, y_2, \dots, y_k \mid x) \propto \exp\left(\sum_{c=1}^{k} y_c\omega + \sum_{c=2}^{d} y_c y_{c-1}\nu + \sum_{c=1}^{k}\sum_{j=1}^{12} y_c x_j v_j\right).$$

- The conditional UGM given x has a chain-structure

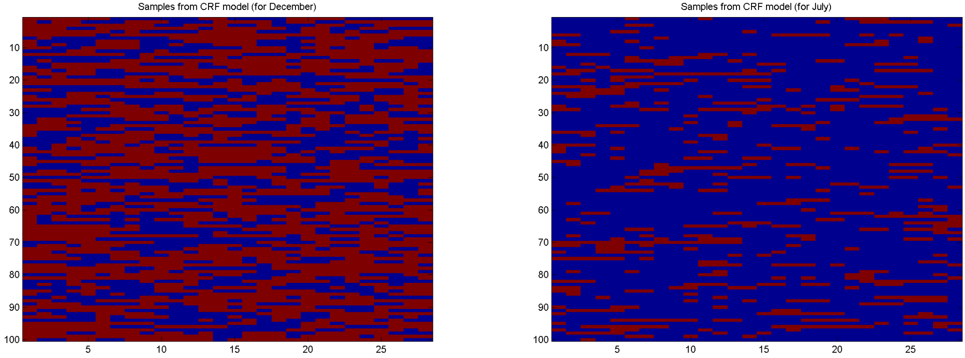
$$\phi_i(y_i) = \exp\left(y_i\omega + \sum_{j=1}^{12} y_i x_j v_j\right), \quad \phi_{ij}(y_i, y_j) = \exp(y_i y_j \nu),$$

so inference can be done using forward-backward.

  - And it's log-linear so the NLL will be convex.

Rain Data with Month Information

- Samples from CRF conditioned on x being December (left) and July (right):



- Conditional NLL is 16.21, compared to Markov chain which gets NLL 16.81.
  - Better than mixture of 10 Markov chains (EM training), which gets 16.53.
    - Probably due to finding global minima when fitting CRF.

Rain Data with Month Information using CRFs

- A CRF model conditioned on month x,

$$p(y_1, y_2, \dots, y_k \mid x) = \frac{1}{Z(x)} \exp\left(\sum_{c=1}^{k} y_c\omega + \sum_{c=2}^{d} y_c y_{c-1}\nu + \sum_{c=1}^{k}\sum_{j=1}^{12} y_c x_j v_j\right).$$

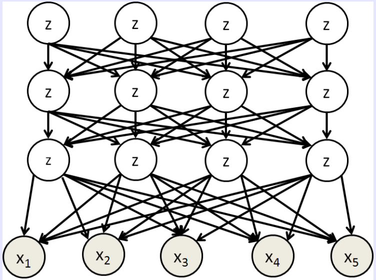
- Comparing this to other approaches:
  - Generative: model $p(y_1, y_2, \dots, y_k, x)$.
    - Have to model distribution of x, and inference is more expensive (not a chain).
    - Also uses known clusters.
    - Learning is still convex.
  - Mixture/Boltzmann: add latent variables z that might learn month information.
    - Have to model distribution of z, inference is more expensive (not a chain).
    - Doesn't use known clusters so needs more data.
    - But might learn a better clustering if months aren't great clusters.
    - Learning is non-convex due to sum over z values.

Summary

- Boltzmann machines are UGMs with binary hidden variables.
  - Restricted Boltzmann machines only allow connections between hidden/visible.
- 3 types of structured prediction:
  - Generative models, discriminative models, discriminant functions.
- Conditional random fields generalize logistic regression:
  - Discriminative model allowing dependencies between labels.
  - Log-linear parameterization again leads to convexity.
  - But requires inference in graphical model.
- Next time: why we are doing everything wrong to make decisions.

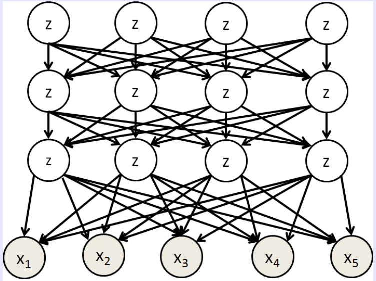
Deep Belief Networks

- Deep belief networks are latent DAGs with more binary hidden layers:



- Data is at the bottom.
- First hidden layer could be "basic ingredients".
- Second hidden layer could be general "parts".
- Third hidden layer could be "abstract concept".

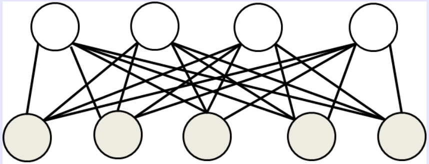
Deep Belief Networks



- If we were conditioning on *top* layer:
  - Sampling would be easy.
- But we're conditioning on the *bottom* layer:
  - Everything is hard.
  - There is combinatorial "explaining away".
- Common training method:
  - Greedy "layerwise" training as a restricted Boltzmann machine.

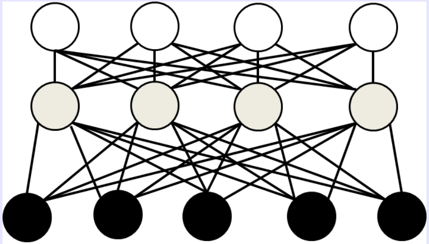
Greedy Layerwise Training of Stacked RBMs

- Step 1: Train an RBM (alternating between block Gibbs and stochastic gradient)



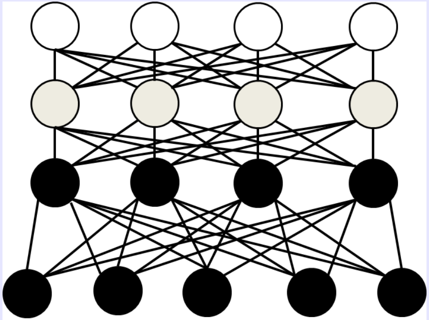
Greedy Layerwise Training of Stacked RBMs

- Step 1: Train an RBM (alternating between block Gibbs and stochastic gradient)
- Step 2:
  - Fix first hidden layer values.
  - Train an RBM.



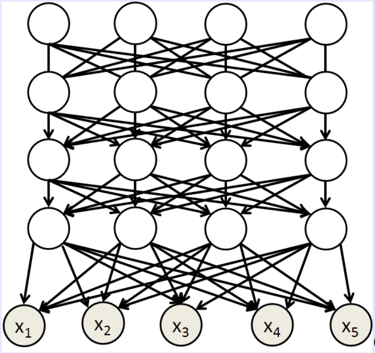
Greedy Layerwise Training of Stacked RBMs

- Step 1: Train an RBM (alternating between block Gibbs and stochastic gradient)
- Step 2:
  - Fix first hidden layer values.
  - Train an RBM.
- Step 3:
  - Fix second hidden layer values.
  - Train an RBM.

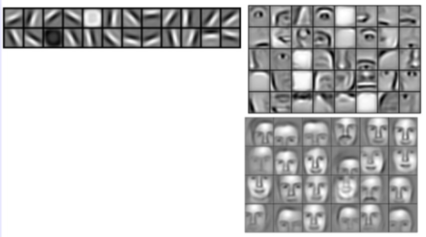


Deep Belief Networks

- Keep top as an RBM.
- For the other layers, use DAG parameters that implement block sampling.
  - Can sample by running block Gibbs on top layer for a while, then ancestral sampling.



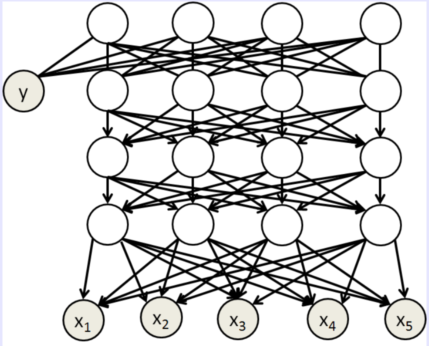
Convolutional:



http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

Deep Belief Networks

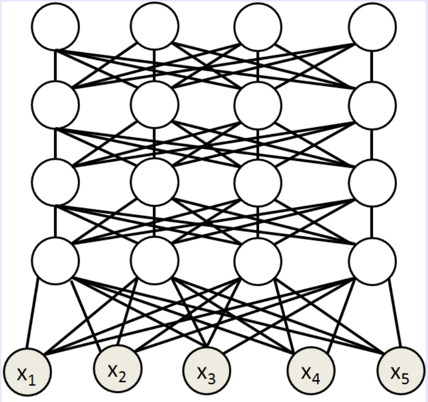
- Can add a class label to last layer.



- Can use "fine-tuning" as a feedforward neural network to refine weights.
  - https://www.youtube.com/watch?v=KuPai0ogiHk

Deep Boltzmann Machines

- Deep Boltzmann machines just keep as an undirected model.
  - Sampling is nicer: no explaning away within layers.
  - Variables in layer are independent given variables in layer above and below.



Deep Boltzmann Machines

- Performance of deep Boltzmann machine on NORB data:

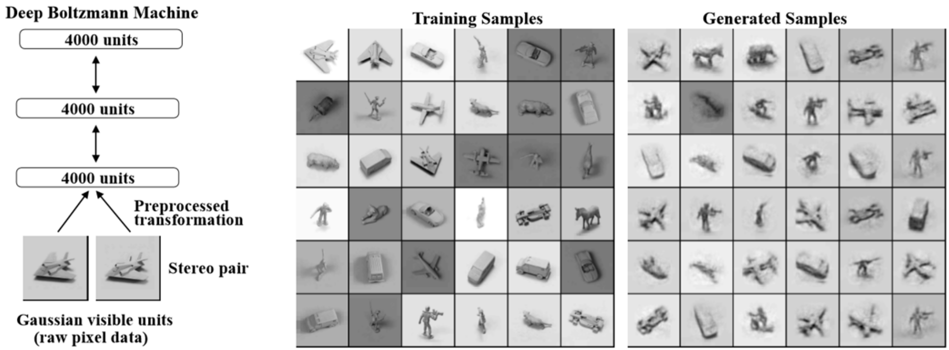


Figure 5: **Left:** The architecture of deep Boltzmann machine used for NORB. **Right:** Random samples from the training set, and samples generated from the deep Boltzmann machines by running the Gibbs sampler for 10,000 steps.

http://www.cs.toronto.edu/~fritz/absps/dbm.pdf

CRF "Product of Marginals" Objective

- In CRFs we typically optimize the likelihood, $p(y \mid x, w)$.
  - This focuses on getting the joint likelihood of the sequence y right.
- What if we are interested in getting the "parts" y_c right?
  - In sequence labeling, your error is "number of positions you got wrong" in sequence.
  - As opposed to "did you get the whole sequence right?"
- In this setting, it could make more sense to optimize the product of marginals:

$$\prod_{c=1}^{k} p(y_c \mid x, w) = \prod_{c=1}^{k} \sum_{\{y' \mid y'_c = y_c\}} p(y' \mid x, w).$$

- Non-convex, but probably a better objective.
- If you know how to do inference, this paper shows how to get gradients:
  - https://people.cs.umass.edu/~domke/papers/2010nips.pdf

Brain Tumour Segmentation with Label Dependencies

- We got a bit more fancy and used edge features x^{ij},

$$p(y^1, y^2, \dots, y^d \mid x^1, x^2, \dots, x^d) = \frac{1}{Z}\exp\left(\sum_{i=1}^{d} y^i w^T x^i + \sum_{(i,j)\in E} y^i y^j v^T x^{ij}\right).$$

- For example, we could use $x^{ij} = 1/(1 + |x^i - x^j|)$.
  - Encourages y_i and y_j to be more similar if x^i and x^j are more similar.

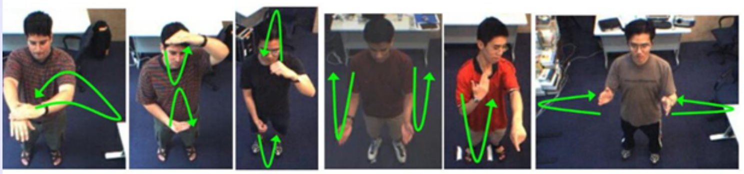
- This is a pairwise UGM with

$$\phi_i(y^i) = \exp(y^i w^T x^i), \quad \phi_{ij}(y^i, y^j) = \exp(y^i y^j v^T x^{ij}),$$

so it didn't make inference any more complicated.

Motivation: Gesture Recognition

- Want to recognize gestures from video:

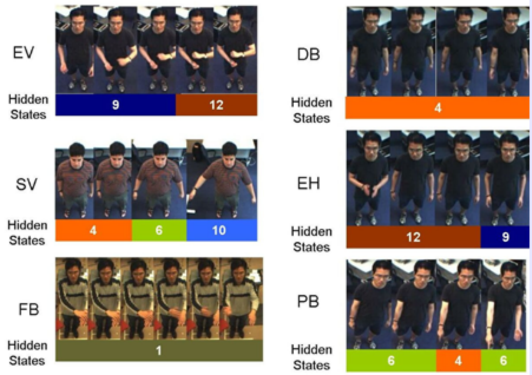


http://groups.csail.mit.edu/vision/vip/papers/wang06cvpr.pdf

- A gesture is composed of a sequence of parts:
  - And some parts appear in different gestures.

Motivation: Gesture Recognition

- We may not know the set of "parts" that make up gestures.

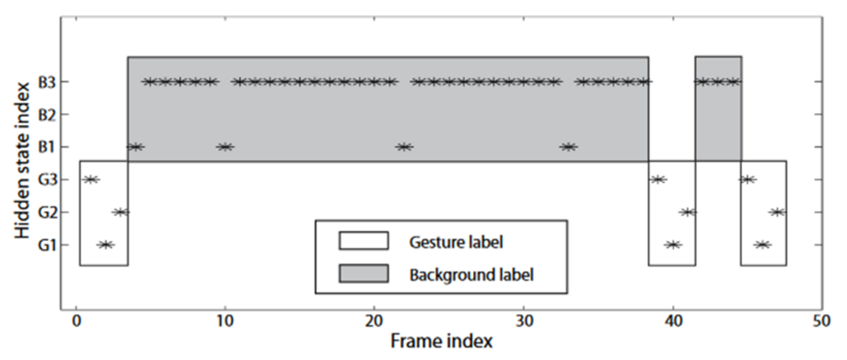


http://groups.csail.mit.edu/vision/vip/papers/wang06cvpr.pdf

- We can consider learn the "parts" and their latent dynamics (transitions).

Motivation: Gesture Recognition

- We're given a labeled video sequence, but don't observe "parts":

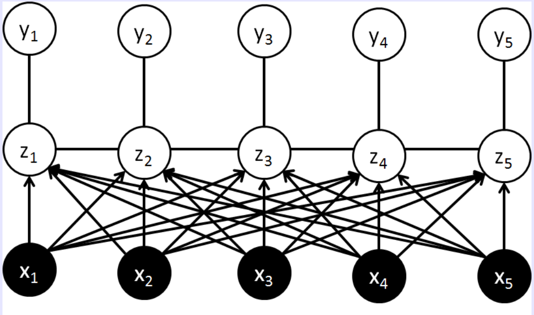


http://www.lsi.upc.edu/~aquattoni/AllMyPapers/cvpr_07_L.pdf

- Our videos are labeled with "gesture" and "background" frames,
  - But we don't know the parts (G1, G2, G3, B1, B2, B3) that define the labels.

Latent-Dynamic Conditional Random Field

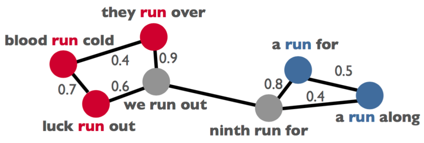
- Here we could use a latent-dynamic conditional random field



- Observed variable x_j is the image at time j (in this case x_j is a video frame).
- The gesture y is defined by sequence of parts z_j.
  - We're learning what the parts should be.
  - We're learning "latent dynamics": how the hidden parts change over time.
- Notice in the above case that the conditional UGM is a tree.

Posterior Regularization

- In some cases it might make sense to use posterior regularization:
  - Regularize the probabilities in the resulting model.
- Consider an NLP labeling task where
  - You have a small amount of labeled sentences.
  - You have a huge amount of unlabeled sentences.
- Maximize labeled likelihood, plus total-variation penalty on $p(y_c \mid x, w)$ values.
  - Give high regularization weights to words appearing in same trigrams:



http://jgillenw.com/conll2013-talk.pdf

- Useful for "out of vocabulary" words (words that don't appear in labeled data).